REMARKS

Claims 1-4 and 7-27 are pending, of which claim 1 is an independent system claim, claim 4 is an independent method claim, with a corresponding independent computer program product claim 15, and claim 24 is an independent method claim. As shown above, claims 1, 4, 15, and 24 have been amended by this paper.

The Office Action rejected each of the pending independent claims (1, 4, 15, and 24) under 35 U.S.C. § 102(b) as being anticipated by TETware release 3.3 ("TETware") as described in the TETware User Guide, Revision 1.2 ("TETware UG") and the TETware Programmers Guide, Revision 1.2 ("TETware PG"). The Office Action rejected each of the remaining dependent claims under 35 U.S.C. § 102(b) as anticipated by TETware or under 35 U.S.C. § 103(a) as being unpatentable over TETware in view of U.S. Patent No. 6,505,342 to Hartmann et al. ("*Hartmann*").[1]

Applicants' invention, as claimed for example in independent system claim 1, relates to selecting and organizing individual test cases, from a binary program module of test cases, for use in testing a computer program to ensure that the program processes as intended. The system includes a binary program module storing a plurality of individually accessible test cases. Each test case comprises a set of instructions for testing a feature of the computer program through a language and format independent interface. At least some of the individually accessible test cases differ from one another in language and format. The system also includes a harness comprising a set of instructions that executes a test case hierarchy on the computer program using the corresponding language and format independent interface of each individually accessible test case in the test case hierarchy. A connector comprising a set of instructions scans the plurality of test cases and extracts those test cases to be used to test the computer program to ensure that it processes as intended. The connector creates a hierarchy of test cases from those that are selected and extracted, and selectively integrates an interface between the test case hierarchy and the harness regardless of the language or format in which the test cases were

[1]As noted in the previous response, *Hartmann* qualifies as "prior art," if at all, under 35 U.S.C. § 102(e). Applicants, therefore, reserve the right to challenge the status of *Hartmann* as a proper reference should such become necessary or desirable in Applicants' view. Accordingly, no argument herein should be construed as acquiescing in the "prior art" status of *Hartmann*, and all such arguments are made simply assuming *arguendo* that the reference qualifies as prior art.

written. The system also includes a processor for executing the one or more test cases, the harness and the connector.

TETware discloses grouping test cases within a test suite. TETware PG, section 2.2. Test suites are organized as directory hierarchies—the top of each test suite directory is known as the test suite root directory. TETware UG, section 5.2.6. All files in a test suite reside below the test suite directory (or in a specified alternate execution directory). TETware PG, section 2.3; TETware UG, section 5.2.6. Test suites are required to include certain files and utilities, such as a build tool (e.g., make), a clean tool (e.g., rm), at least one test scenario file, etc. TETware PG, section 2.5.

A test scenario is a list of invocable components from a test suite that are processed during a particular TETware invocation. TETware UG, section 2.2. Within a scenario file, a test case name may appear by itself or be attached to a directive that describes how the test case should be executed (sequentially, in parallel with other test cases, repetitively, remotely, etc.). TETware PG, section 4.2.4.3; TETware UG, section 5.3.2.2. Test case names are interpreted relative to the test suite root directory or alternate execution directory, depending on the mode of operation. TETware UG, section 5.3.2.4. Section 5.3.2.5 of the TETware UG and section 4.4 of the TETware PG present some simple examples of test scenarios. Example test cases names listed in these scenarios follow the directory convention explained above (e.g. "/tset/tc1" and "/ts/tc1").

The test cases in a test suite are processed by a Test Case Controller (tcc), based on a chosen mode of operation (e.g., build, execute, clean up). TETware PG, section 2.5. In build mode, the tcc translates source test cases into executables, in execute mode the tcc loads and executes test cases, and in clean mode the tcc removes unwanted files. TETware PG, section 3.2. Each test case is an executable program. TETware PG, section 2.2. When a test case uses one of the TETware language specific APIs, its execution is supervised by a Test Case Manager (TCM). TETware PG, section 2.4.2. The TCM is not a separate program, but instead is linked with user-supplied test code and the API library to produce an executable test case. *Id.* There is a separate TCM module for each API that is supported by TETware. *Id.* For example, TETware includes a C TCM and a C++ TCM. TETware PG, section 2.4. Test cases are written to a specific language binding (C, C++, Shell, Korn Shell, etc.). TETware PG, sections 8, 9, 10, and 11.

The Office Action asserts that TETware's test suite corresponds to Applicants' claimed binary program module. As explained above, however, a TETware test suite is simply a directory hierarchy of test cases, not a binary program module. In TETware, each test case within a test suite's directory hierarchy is a separate executable. TETware discloses test cases interfacing with TCMs, but these interfaces are language specific (otherwise there would be no need for separate C, C++, Shell, Korn Shell, and Perl TCMs). TETware PG, section 2.4.3; TETware UG, section 2.4. While TETware may use a single test case controller for calling all test cases, it does so by keeping test cases written in different languages as separate executables, with a language specific interface, rather than through a binary program module of test cases with language and format independent interfaces.[2] Accordingly, among other things, TETware fails to teach, suggest, or motivate a binary program module storing a plurality of individually accessible test cases, each comprising a set of instructions for testing a feature of a computer program through a language and format independent interface, at least some of the individually accessible test cases differing from one another in language and format, as claimed for example in independent system claim 1. Similar limitations can be found in each of the pending independent claims.

Applicants note for the record that any remaining assertions and rejections for the independent or dependent claims are now moot.[3] Accordingly, Applicants reserve the right to challenge those assertions and rejections, should such challenges become necessary or desirable in the future. For at least the foregoing reasons, therefore, Applicants respectfully submit that the cited art does not anticipate or make obvious Applicants claimed invention. Consequently, Applicants believe that all pending claims are in condition for prompt allowance. In the event that the Examiner finds any remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney.

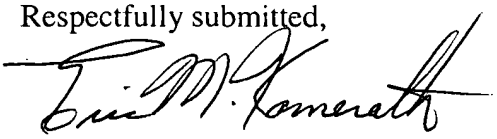
---

[2]The Office Action asserts that different test cases inherently differ from one another in format. As support for this proposition, the Office Action states that at a low level, they comprise different bit patterns, and at a higher level, they comprise different instructions or parameters. Applicants submit however, that this logic merely shows the obvious—that different test cases differ. The logic ignores that, independent of test case content, test cases also may have different formats (e.g., from being written in different languages), which as Applicants note, may lead to increased testing and maintenance overhead. See, e.g., Specification, p. 2, l. 23 – p. 3, l. 17.

[3]Applicants have corrected the grammatical informalities pointed out in the Office Action with respect to claim 1 and claim 24.

Dated this 4th day of February, 2004.

Respectfully submitted,



RICK D. NYDEGGER
Registration No. 28,651
ERIC M. KAMERATH
Registration No. 46,081
Attorneys for Applicant
Customer No. 022913

DMS0000005565V001